Amendments to the Claims

Please amend claims to be as follows.

1. (currently amended) A method of processing a data packet, the method comprising:
receiving the data packet at a network device;
determining whether a multiple-key decision cache is hit by the data packet;
applying at least one cached action if the decision cache is hit; [[and]]
processing the data packet using software routines if the decision cache is missed;
determining whether action performed by the software routines is programmable into the multiple-key decision cache; and
programming a new entry into the multiple-key decision cache if the action performed is programmable,
wherein the new entry indexes into the multiple-key decision cache, and programming the new entry does not involve storing the data packet.

2. (original) The method of claim 1, further comprising, prior to determining whether the multiple-key decision cache is hit:
determining whether hardware circuitry of the network device is capable of processing the data packet; and
processing the data packet using the hardware circuitry if the hardware circuitry is determined to be capable.

3. (canceled)

4. (currently amended) The method of ~~claim 3~~ claim 1, wherein a hash value relating to multiple fields in the data packet is used in programming the new entry.

5. (original) The method of claim 1, wherein determining whether the multiple-key decision cache is hit comprises:
generating a hash value from multiple fields in the data packet; and

using the hash value generated to index into the multiple-key decision cache.

6. (original) The method of claim 5, wherein the hash value is generated by applying an exclusive-or operation to a source IP address and a destination IP address in the data packet.

7. (original) The method of claim 5, wherein
   if the hash entry is valid in the multiple-key decision cache, then determining whether pertinent fields of the data packet exactly match corresponding fields of the entry; and
   if the pertinent fields exactly match, then providing a result that the decision cache is hit.

8. (original) The method of claim 1, wherein search keys for the decision cache include source and destination IP addresses.

9. (original) The method of claim 8, wherein the search keys further include an inbound VLAN identifier.

10. (original) The method of claim 1, search keys for the decision cache include source MAC addresses.

11. (original) The method of claim 1, wherein if a modification of a pertinent table is detected, then the decision cache is cleared and populated if possible.

12. (original) The method of claim 11, wherein the pertinent table comprises a table from a group of tables including a network address translation (NAT) table, an access control list (ACL), a network layer 3 forwarding table, and a network layer 2 forwarding table.

13. (original) The method of claim 1, wherein if a modification of a forwarding table is detected, then the decision cache is cleared.

14. (original) The method of claim 1, wherein if a modification of a pertinent table is detected, then a corresponding entry in the decision cache is cleared.

15. (original) The method of claim 1, wherein if a modification of a pertinent table is detected, then a corresponding entry in the decision cache is updated.

16. (currently amended) A network apparatus comprising:
    a plurality of ports configured to receive data packets;
    software routines configured to process the data packets;
    a multiple-key decision cache including multiple key fields and action(s) corresponding thereto; and
    logic configured to determine whether the multiple-key decision cache is hit by a data packet, to apply at least one cached action if the decision cache is hit, and to process the data packet using the software routines if the decision cache is missed, and further to determine whether action performed by the software routines is programmable into the multiple-key decision cache, and to program a new entry into the multiple-key decision cache if the action performed is programmable, wherein the new entry indexes into the multiple-key decision cache, and programming the new entry does not involve storing the data packet.

17. (original) The apparatus of claim 16, further comprising:
hardware configured to rapidly process a subset of the data packets; and
hardware logic configured to determine whether the hardware circuitry is capable of processing the data packet, and to process the data packet using the hardware circuitry if the hardware circuitry is determined to be capable, prior to determining whether the multiple-key decision cache is hit.

18. (currently amended) A method of processing a data packet, the method comprising:
receiving the data packet at a network device;
determining whether hardware of the network device is capable of processing the data packet;
if the hardware circuitry is determined to be capable, then processing the data packet using the hardware;
otherwise, determining whether a decision cache is hit by the data packet;
applying at least one cached action if the decision cache is hit;
processing the data packet using software routines if the decision cache is missed;
determining whether action performed by the software routines is programmable into the multiple-key decision cache; and
programming a new entry into the multiple-key decision cache if the action performed is programmable<u>, wherein the new entry indexes into the multiple-key decision cache, and programming the new entry does not involve storing the data packet</u>.

19. (original) The method of claim 18,
wherein a hash value relating to multiple fields in the data packet is used in programming the new entry,
wherein if the hash value matches an entry in the multiple-key decision cache, then determining whether pertinent fields of the data packet exactly match corresponding fields of the entry, and
wherein if the pertinent fields exactly match, then providing a result that the decision cache is hit.

20. (original) The method of claim 19, wherein the hash value is generated by applying a hash function to source and destination IP addresses.